IEN 32

Section 2.3.3.12

Postel

CATENET MONITORING AND CONTROL: A MODEL FOR THE GATEWAY COMPONENT

John Davidson

Bolt Beranek and Newman, Inc.

IEN #32

Internet Notebook Section 2.3.3.12

April 28, 1978

Postel

IEN 32

Section 2.3.3.12

Catenet Monitoring and Control:
A Model for the Gateway Component

1. INTRODUCTION

At the last Internet meeting, some concern was expressed that we don't have a real "model" for what a gateway is, what it does, and how it does it, and that without such a model it is somewhat dificult to describe the kinds of activities which should be monitored or controlled by a Gateway Monitoring and Control Center (GMCC). To respond to that concern, we have written this note to express our recent thoughts about a gateway model. Although the note centers primarily around the topic of a gateway model, we have also included a few thoughts about possible models for the other components of a general internetwork structure.

The note proceeds as follows. In Section 2 we try to establish a reason for wanting a model of a given internetwork component, and present a brief overview of the potential benefits of Monitoring and Control. This presentation is largely pedagogical since it is assumed that this document will, for a while at least, be the only introduction to the topic available.

In Section 3 we discuss the fundamental kinds of activities which must be performed by any internet component if it is to participate in Monitoring and Control. This section establishes motivation for some of the mechanisms discussed in the rest of the paper.

In Section 4 we discuss the roles which the hosts, local Packet Switching Networks (PSNs), and the Gateway Monitoring and Control Centers (GMCCs) may have to play in Monitoring and Control for the internet.

Then, in Section 5, we finally begin to discuss the principle characteristics of a possible gateway model. We examine first a list of practical constraints which influence the way in which the model is being designed, and then, suitably constrained, we begin the task of developing the model itself. A complete modelling has not yet been performed, and is likely to take quite a while to complete. Section 5, however, gives a suitable indication of the way in which the model will be developed, and of the alternative interpretations available to gateway designers who pattern their implementations after the model.

2. CONTEXT F OR MODELING

It is assumed that gateways exist to make communication possible between hosts on different packet switching networks. In this regard, they actually serve to make a single network out of several diverse, disjoint networks. Thus gateways can be said to form the nodes of a super-network, called a Catenet, whose "links" are the individual packet switching networks, and whose "hosts" are simply the individual hosts of the constituent PSNs. As the nodes of this super-net, the gateways will be responsible in part or in whole for tasks like routing, fragmentation, flow control, reassembly, retransmission, and perhaps data transformation (e.g. encryption/decryption), access control, and even authentication.

We can assume that there are benefits to be derived from having the Catenet operate in a coordinated fashion. However, coordination is not achieved by default, because the Catenet is being constructed in pieces, with each piece potentially under the control of a distinct administrator, each gateway implemented in a unique processor, and each program conforming to a different programmer's view of the workings of a gateway.

The Internetworking group brought together by ARPA is serving as the coordinating authority for the development of many of the components of the Catenet. To make the important administrative and technical decisions associated with Catenet construction, however, this group must be provided with technical inputs. Many of these inputs can come from theoretical studies, protocol investigations, and pre-construction experiments, modeling, and simulation during Catenet development. But the most important source of technical inputs will ultimately be the Catenet itself. If the true operational characteristics of the net can be readily (and continually) determined, then the administrative issues associated with Catenet performance (questions like "why is the throughput not as predicted", "what components are proving unreliable", "should the net be expanded or reconfigured", etc) can be adequately resolved by appeal to the recorded statistics; collection and recording of these operational statistics form a part of what we refer to as Catenet "Monitoring". Also, since the Catenet will sometimes be the "target" of the Internet group or ARPA administrative policy, then, insofar as policy affects the operation of the net (e.g. such "policy" decisions as "failed components should be dumped, reloaded, and restarted ASAP"), technical tools must be provided which help implement the policy; these kinds of tools form a part of what we refer to as Catenet "Control". (Some readers may prefer to think of this as "Coordination" rather than "Control" which unfortunately may carry other undesirable connotations.)

In order to be able to Monitor and Control the operation of the Catenet in accordance with administrative policy, in the presence of the diverse component implementations, some model for the network as a whole should be developed. Futher, models for each of the component types (gateways, packet switching nets, hosts) should be developed. These model implementations should then be instrumented in a way that makes it possible to accumulate the technical inputs and to effect the operational policy desired by Catenet administrators. If the model implementations are appropriately general, then it is reasonable to dictate that individual implementations adhere to these basic models.

BBN is currently pursuing the development of a model for the gateway component of the Catenet. In particular, we are trying to define how the model should be instrumented to provide the appropriate kinds of Monitoring and Control capabilities. Most of the capabilities we think are needed are similar in function to the capabilities already developed for the ARPANET and its Network Control Center, NCC. We are proposing, and in fact have actually begun, the construction of a Gateway Monitoring and Control Center (GMCC) patterned after the NCC (and currently sharing resources with it, since the NCC is already accessible to internetwork traffic and since the scope of the current GMCC is at this time quite modest).

3. FUNDAMENTALS OF MONITORING AND CONTROL

We assert that all of the Catenet components should be properly instrumented in software (and if necessary, in hardware) to measure the service which the Catenet as a whole provides, and to enhance the maintainability of the net as a whole. The instrumentation should provide us with all the mechanisms required to perform

Performance Monitoring

Event Recording

Functional Testing

Component Maintenance

Here, by Performance Monitoring, we intend that the status of Catenet components (both the binary "working/not-working" indication and the status of internal operational components) be made available to the GMCC. This will require that the Catenet components have a mechanism for communicating periodic status reports and instantaneous error reports "back" to the GMCC. This mechanism may or may not require that the reports from the various net components be synchronized in order to enable the GMCC to obtain a "snapshot" of the network as a whole; if synchronization is required, a synchronizing mechanism will be required as well. It is also undetermined whether a protected path (e.g. one using encryption to prevent spoofing) is required for communicating this information through the Catenet.

There should also be a mechanism by which the GMCC can request performance data from a Catenet component in the case of non-recurring measurements. The set of monitoring mechanisms installed in any particular component may differ from the set installed in any other component, depending on the granularity of measurement which is desired in each case.

By Event Recording, we intend that the raw statistical data on, say, the number of messages or packets passing through a given component be made available to the GMCC in a standard way. Not only must there be a standard way of collecting the event statistics, but there must also be a way for a designated authority like the GMCC to reset the event counters, say, or turn on or off the event recording mechanisms. We shall have more to say on what events are significant in a subsequent section.

By Functional Testing we intend that the GMCC be able to direct the activities of the Catenet components either directly (by commanding performance of some task like message generation) or indirectly (by sending, or directing other components to send, messages through a given component) in order to exercise component mechanisms for error analysis or load testing. A useful mechanism in the ARPANET is the ability to isolate failed hardware components by forcing a loopback under software control at each of the component boundaries. The analogue of this scheme in the Catenet is probably simpler, since the boundaries are logically in software (e.g. in the gateway software in cases where a local PSN or another nearby gateway is being tested) or associated with the local PSN which may have reasonably flexible control of the interface which connects it to a network host or to a gateway. Looping performed within a component should also be possible.

To make use of these testing facilities, we need the ability to generate artificial traffic (and to discard it) and to reflect it (or turn it around) as required. Reflecting messages at gateways can, for example, give a measure of the throughput of Catenet links.

By <u>Component Maintenance</u>, we intend that the GMCC have the ability to coordinate, and in some cases perform, the analysis of failure for failed components, the restoration of failed components, the institution of program fixes, and the distribution of new releases. It is not clear that Component Maintenance can be the responsibility of just a single GMCC, of course, but if it is to be the responsibility of any GMCC-like component, then the mechanisms by which the failed component is to be handled, and how it is to be of use in the maintenance activity, should be adequately modelled for each component in question. In this regard, we see a potential need for incorporating mechanisms for self diagnosis in the component models, for enabling the GMCC (or some other network host in conjunction with or in place of the GMCC) to rea@ and write arbitrary locations in a component's memory, etc.

Note too that the gateway programs will be provided by various implementers. These implementers may in some cases be willing to allow a given GMCC to handle reloads and restarts of their component when it fails. Both the implementer and the GMCC staff may have to be involved in debugging the component if the gateway's model debugging facility (which presumes the use of a GMCC) is all the original implementers have for accessing their component remotely. It might prove useful for the GMCC to be able to copy the contents of a failed component into a file for later inspection by the original implementers in case they are unavailable to copy the contents themselves at the time of malfunction.

4. MODELS FOR THE PRINCIPLE CATENET COMPONENTS

This note is principally about gateway modelling. However, we have asserted throughout, the need to model the other components of the general Catenet as well, so that we can use them in Monitoring and Control applications where they are needed and where they can be useful. Here we describe briefly the goals we have for modelling the other components.

4.1 The GMCC

The functions of a GMCC should be able to be performed by any host in the composite net. In view of this, a high level description, or model, of the way a GMCC operates should be created. Both the GMCC program(s) and its data base(s) should be described in a way which allows Catenet users to reproduce the GMCC functions (this basically requires coding of the GMCC programs in a high order language) and to interrogate or duplicate the accumulated data base(s) as required for their own special purposes.

Because each gateway, host, or local PSN component of the composite Catenet is potentially the property of a distinct administrative authority, it is conceivable that each might actually be monitored or controlled by a distinct GMCC. This would not necessarily be the best arrangement for purposes of overall net maintainability, but nonetheless must be allowed for. What is more likely, however, is that a given administrator will give permission to some approved Catenet GMCC to perform the Monitoring and Control activities associated with Performance

Monitoring, say, or with Event Recording, but reserve for itself the ability to perform the activities associated with Functional Testing and Component Maintenance. In this case, the Administrator's host will have to understand and be able to duplicate the model GMCC functions, and the Catenet component will have to know to respond to one or the other GMCC depending on the function being requested. Since different authorities might exist for each different function, this capability should be modelled. Further, the mechanism for changing the name or address of the various designated authorities should also be modelled. Then the fact that each Catenet component knows the name of the authority designated to perform a given function makes it possible to restrict arbitrary hosts from abusing their ability to emulate a GMCC.

4.2 The Hosts

Members of the ARPANET NCC staff have asserted that "a key factor in network maintainability is the centralization of the responsibility for providing adequate user service. Since service is best defined at the man/machine interface, a significant gain in maintainability would come about if the user interface were completely at the man/machine boundary. By including a host within the sphere of responsibility of network maintenance, there could be more uniform and speedier resolution of problems within that host. Since the network design allows for dissimilar node programs, not much additional complexity is required to maintain a set of dissimilar service hosts. (Thus) the scope of the network should be expanded to providing user services with corresponding benefits for both unified design and maintainability."

This may be an extreme position, and may not actually be as easy as anticipated by the NCC staff, but nevertheless it is a position which has at least some validity, especially in view of the fact that gateways are themselves just hosts, and much of the modelling performed for gateways can thus be applied to other general purpose hosts.

Consider what Monitoring and Control might be possible if some small component of the host's network software, such as the TCP program, were instrumented to allow Performance Monitoring, Event Recording, etc. under control of a GMCC. The instrumentation would simply provide that the TCP keep track of its events of interest and arrange for them to be made available in a convenient way to some other protocol module (perhaps) for transmission to the GMCC. In addition, if there is to be Control of a TCP, some internal means should be provided for a process to direct certain actions of the TCP (for example the resetting of accounting statistics).

Certain other capabilities might also prove useful. The TCP should be able to report to the GMCC any errors it observes in packet formats, packet delivery, etc. so that host personnel with a reliable TCP implementation need not be concerned about error analysis for newly added, undebugged hosts, say. The GMCC is certainly in the appropriate position to be able to correlate abnormal TCP interactions with Catenet component outages and be able to explain the abnormal behavior to the host via messages to the TCP. The TCP should be able to be instructed to discard certain messages or to echo them back to their source. It should perhaps be able to timestamp and trace various messages. These kinds of activities would all be possible given an appropriate and uniform instrumentation of the various TCP implementations.

4.3 The PSNs

The links of the Catenet are the local PSNs. Unlike usual network links, these links are equipped with a processing capability in the form of their own node computers or their own NCC hosts, etc. Whatever form this processing capability takes, it can presumably be made to communicate with other Catenet components using the protocols for GMCC-to-component and component-to-GMCC communication. The local PSNs should be able to report errors in interfacing which occur at the PSN/gateway interface; they can also report gateway ups and downs as they observe them; they might be instrumented to assist in tracing and looping of messages sent to or through them; they could keep track of the length of gateway outages; and since the PSN is the only component with hardware (in most cases) connections to the gateway and host components, it can perhaps help in the restart or reload of these components. The techniques for performing any of these activities should be carefully and completely modelled.

5. A MODEL FOR THE GATEWAY COMPONENT

The principle thing we expect a gateway to do is to perform message routing; a suggested routing mechanism is presented in IEN No. 30, "Gateway Routing: An Implementation Specification", by Strazisar and Perlman. Beyond this, there are several secondary activities in which the gateway must play a role, and the large majority of these can be clustered under the general heading of Monitoring and Control. These are the activities we are most concerned with here. As discussed earlier, the gateway component, like any other component, should be instrumented to include mechanisms which allow it to cooperate with a GMCC in providing Performance Monitoring, Event Recording, Functional Testing, and Component (i.e. gateway) Maintenance. It is the role of the model to identify the mechanisms which should be used within the gateway to provide these various functions.

5.1 Considerations Affecting the Design

The intent of this section is to give some insight into the process by which the model for the gateway component will be developed. There are a number of fundamental considerations which should be stated beforehand because of their impact on the way we want to think of gateways as an entity and thus on the way we think of the necessarily composed. The first of these is that a gateway should be considered to have a net-independent part and one or more net-dependent parts. The net-independent part should be considered the heart of the gateway -- the place where the common functions of routing, flow control, etc. are actually performed; this part is hopefully divorced from considerations of the networks to which the gateway is attached. The net-dependent parts on the other hand may all be different, since the networks in most cases will be different, but each should have the same eessential structure: there will be modules which gate messages to and from the attached net, and modules which append or remove local headers to or from internet (and other) messages, etc. One of the challenges for the model designer and gateway implementer is to carefully design the boundary between the net-dependent and net-independent functions.

The gateway model should be able to accommodate more than one type of gateway implementation. That is, it should accurately describe or apply to implementations such as:

- the conventional gateway. This is a single processor performing gateway functions only and connected to only two nets.

- a two- or multi-processor gateway. This is a distributed implementation of the gateway, such as the "half-gateway" model once considered; the mechanisms used by the separate processors to communicate with each other should not affect the model design.

- gateways within general purpose host processors where the gateway program is just one of several that may want access to the network.

- gateways connected to three or more networks.

- gateways using only a single net interface. Such an arrangement might result if, for example, a single medium were used by two different nets. readdressing, say, ...

- both big and small (in terms of power, size, cost, capability) gateways (including very simple - perhaps purely hardware - implementations).

- existing ARPANET gateways.

- existing othernet (sic) gateways.

- the gateway module which sits between a host TCP, say, and the network, and whose job it is to select a destination gateway consistent with the destination host.

- a gateway with two or more interfaces to a single network.

- a gateway which is (either always or sometimes) unwilling or unable to participate in Monitoring and Control exercises.

- gateways which are responsible for access control or authentication. The need for these special functions may impact the form of certain mechanisms proposed for use in the model implementation.

- gateways which need to perform fragmentation or reassembly of encrypted data messages, or which need to be able to understand special packet formats associated with secure data transmissions.

- gateways which may without warning turn into "one-way" paths only for such applications as military EMCON.

5.2 The Beginnings of a Model

There are several kinds of information which the gateway should be able to exchange with the GMCC in support of its Monitoring and Control activities. Among them are:

Administrative Information

This is information that identifies the gateway uniquely among all the components of the composite Catenet. To get a proper picture of the net at any given time, a GMCC would like to be able to discern, among other things,

the computer type
the memory size
the gateway characterization (conventional, ARPANET, ...)
the Administrator's name, address, phone number
the program size, release number, release date and time

the addresses of hosts serving as GMCCs
the names of connected nets, etc.

Information such as this may best be sent unsolicited to a special service host when a gateway first comes up on the net after some outage; interested experimenters could then retrieve the information from the host much as is currently done in the ARPANET for retrieving Host status information.

Measurement Information

This information is simply the collective set of statistics accumulated by the gateway during its operation. They reflect the processing performed by the gateway in servicing internet traffic.

Monitoring Information

This is primarily the "up/down", "up for how long", "planned outages", "recent crash explanation", "net interface reset", etc. kinds of information which dictates to the GMCC the status and health of the gateway component.

Control Information

This is either the information sent by the GMCC to cause the gateway to enter a test, reload, or, restart mode, or the information sent by the gateway to the GMCC to report the results of component testing, to dump some of its memory, etc.

Debugging Information

Patterned after a general purpose time sharing host's DDT program which has complete control over the execution of subservient programs, the information associated with debugging includes commands to read or write a cell, to set or reset (pseudo) breakpoints, to search memory, etc. and the responses to these commands. Two kinds of DDTs may have to be accommodated in any given gateway implementation, one to be used by experimenters, say, and one, like XNET, to be used during initial debugging.

Descriptive Information

This is the information which conveys to the GMCC the list of capabilities possessed by the gateway; it includes a listing of the kinds of information collected and reported by the gateway, a characterization of queue capacities, a list of settable operational parameters, a description of the histograms maintained by the gateway, a list of the protocols supported, etc. It responds to the GMCC's questions about "what can you do", "how much can you do", etc.

Experimental Information

This is the information associated with the manipulation of the component by experimenters. It is in part descriptive (experimenters can ask what experiments are supported, what parameters are allowed, what range on parameters is accepted, etc.), in part control (they can request initiation of an experiment), and in part measurement (they can request operational statistics associated with the experiment), but it seems reasonable to distinguish it as distinct from these other operational aspects insofar as possible; not all gateways which provide descriptive, control, and measurement information will also support experimental use.

Accounting Information

This information is basically the set of raw statistics which should be used by an administrator for charging for gateway utilization. It is reasonable to distinguish this information from pure measurement information since it is not necessarily of interest to a GMCC worrying about functional capabilities, and will likely have to be reported to some special host rather than a general purpose community GMCC.

Operational Information

This is included here just as a reminder that in addition to manipulating all the information associated with Monitoring and Control activities, the gateway will also want to occassionally handle internet messages, routing messages, and the rest of the information that is its "reason for being" in the first place!

Note that it is possible to consider that each individual kind of information is associated with a particular kind of "event" which occurs within a gateway. Thus we may have Measurement events, Monitoring events, and even Administrative events within a functioning gateway. It is also the role of the gateway model to specify how these events are to be recognized, recorded, reported, caused, prevented, suspended, continued, etc.

At least three notions are central to our discusionss at this point. First, we have the four basic functions that we have discussed in detail before: Performance Monitoring, Event Recording, Functional Testing, and Component Maintenance. From a suitable, high level external viewpoint, these are the functions that the gateway is involved in. Second, we have the different kinds of information which must be recorded by the gateway and transported between the gateway and the GMCC. Each different kind of information implies possibly a distinct formatting requirement, distinct collection mechanism, etc. Finally, there are the several different mechanisms which must exist inside the gateway that can be used to collect, record, and report the different kinds of information. The most apparent mechanisms which exist in the gateway implementations are processes. Processes are the addressable resources which carry on dialogues with the GMCC and with each other in some cases. In addition to the distinct processes, there are other mechanisms which we will just label as "routines". Routines are best thought of as utility functions which may be invoked by any of the gateway processes to help each get its collecting, recording, and reporting done. An example of a utility routine might be one which formats a message according to gateway-to-GMCC protocol specifications and adds it to a queue of messages to be sent. Examples of processes are

- the monitoring process which generates the periodic and spontaneous reports to the GMCC describing the operational status of the gateway.

- the measurement process which delivers cumulative statistics to the GMCC.
- the echoing process which returns all received messages to the source from which they originated.
- the memory transport process which moves portions of the gateway memory (programs or data) to or from the GMCC.
- the terminal handling process which excanges ASCII character streams between the gateway's terminal (if one is present) and a specified internetwork source or destination.
- the debugging process.
- the message generator process.
- etc.

It should be obvious that processes do not deal one-to-one with the kinds of information we discussed above. A given process, such as the measurement process, may be used to report the cumulative statistics of each of several kinds of information. Alternatively, it may take more than one process to deal with all the information of any particular kind; for example experimental information as discussed above. And it is certainly likely that multiple distinct processes will have a need to share a single common routine whenever their processing requirements are identical; for example, tracing of messages sent from the GMCC to the debugger, to the echoer, or to the terminal handler should be done by having each process (conceptually) invoke the single trace mechanism. It may also be the case that two processes can be cascaded for the purpose of combining different kinds of information into a single net message.

5.3 A Sample Modeling

We will develop the gateway model in terms of a general purpose host's implementation, since this allows inclusion of as much mechanism as may be useful for the more general implementation. The figure below shows the principle components of the input handler for a net-dependent part of a general purpose gateway.

First there is a hardware component which represents the physical interface between the network and the gateway processor. Obviously this interface will be different for different nets and for different processors, but as a model component should always correspond to some real chunk of hardware in any implementation.

Second, there is a piece of code which serves to control the input portion of the net interface hardware. Its basic function is to effect the reading of messages from the network into gateway memory. In the process, it may perform intermediate parsing, do checksum and consistency processing, etc.

Third, there is a message queue where unparsed messages reside after they have been received by the net interface routine but before they have been processed by any other gateway routine. This queue may be implemented in any of several ways, and may only have room for a single message in some implementations. (A "higher" level routine may perform queue management in this case, using a different data structure.)

A MODEL INPUT HANDLER FOR ONE NET-DEPENDENT INTERFACE

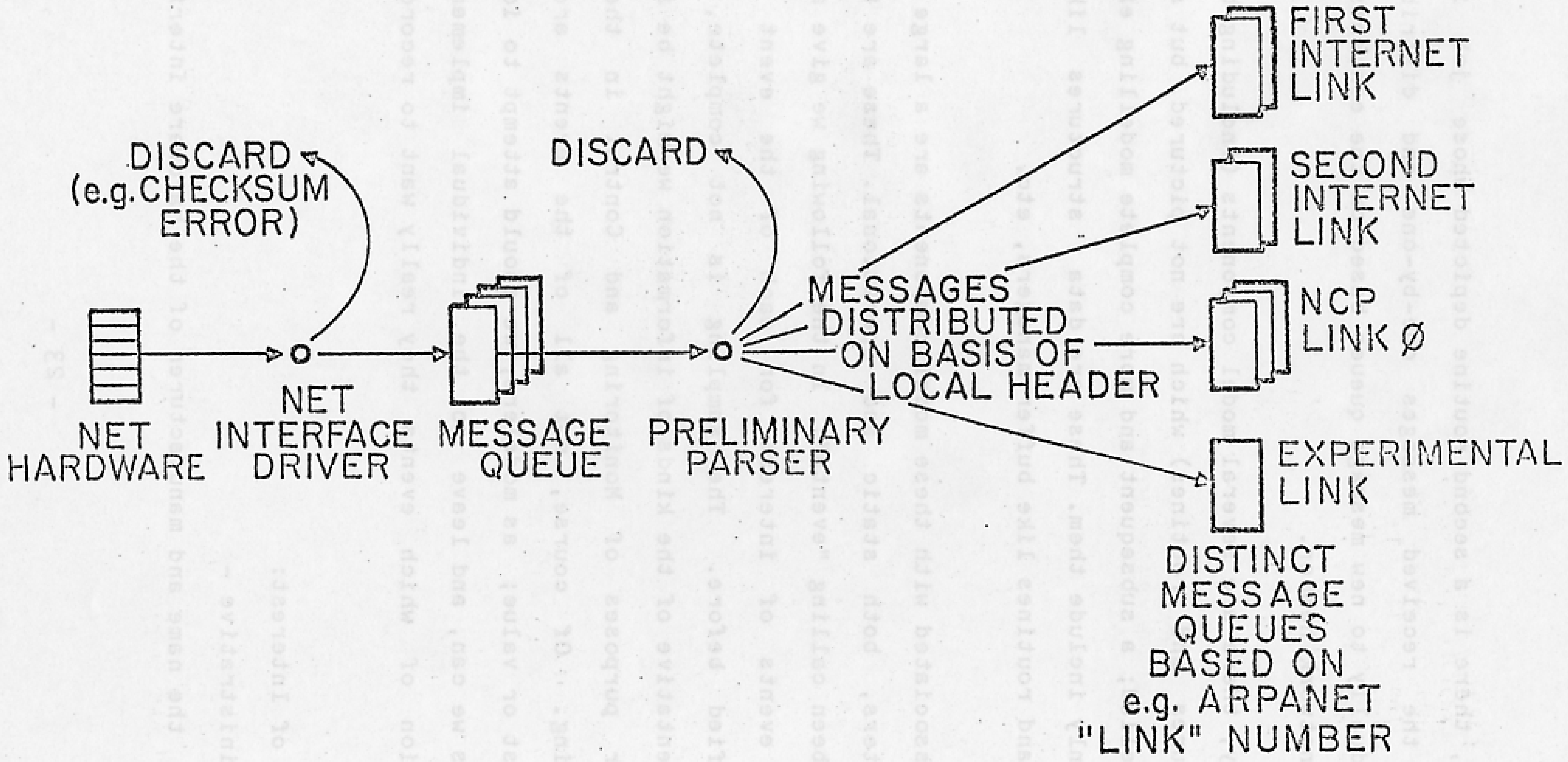

Figure 1

Fourth, there is a second routine depicted whose job it is to parse the received messages one-by-one and distribute them individually to new message queues based on the contents of their local network header.

Finally, there are several model components (including both data structures and routines) which are not pictured, but still must be modelled; a subsequent and more complete modelling effort will certainly include them. These are data structures like buffer pools and routines like buffer managers, etc.

Associated with these model components are a large number of parameters, both static and operational. These are the things we've been calling "events". In the following we give a sampling of the events of interest for each of the event types we identified before. The sampling is not complete, but it is representative of the kinds of information we might be interested in for purposes of Monitoring and Control in the gateway modelling. Of course, not all of the events are of equal interest or value; as modelers, we should attempt to identify as many as we can, and leave to the individual implementers the selection of which events they really want to record, report, etc.

Events of Interest:

Administrative -

the name and manufacturer of the hardware interface

Measurement -

a histogram of log[base 2] message sizes
message counts for each distinct local header type

Monitoring -

cumulative uptime
number of interface errors

Control -

reset counters
place a message on the input queue

Debugging -

read the hardware status register

Descriptive -

Fan out for local headers
queue size
maximum message size

Experimental -

discard every second message at the interface

Accounting -

total bits received at the interface

5.4 Continuing the Sample Modeling

In this section we continue the sample model introduced above by showing how certain of the data paths might be extended to account for subsequent message processing. It should be easy to identify the events of interest in this extension given a little thought. Subsequent attempts at modelling will enumerate these in detail. Note that there are probably several alternative ways of depicting these later stages of message processing; this fact is compounded by the fact that this is the point in message processing where the net-dependent/net-independent boundary may be crossed. Further discussion of alternatives to this part of the model is postponed to the section entitled "Issues".

Figure 2 shows the extension to the model. It begins where the previous figure left off. First, note that at this point we have separated the various types of messages arriving at the net interface into unique queues according to indicators in the local header. For this figure we will follow only a single path -- the one followed by internet messages which carry the normal internet traffic. These internet packets are removed from their queue by a routine which separates them again, this time according to the version bit, into a queue of messages which employ the previous internet formatting conventions and a queue of messages which employ the current conventions.

From this latter queue, the messages are sent to another routine whose job is to initiate the option processing. In Figure 2, we have represented the options as subroutines without further elaboration; subsequent versions of the model should provide the elaboration.

CONTINUATION OF THE INPUT HANDLER

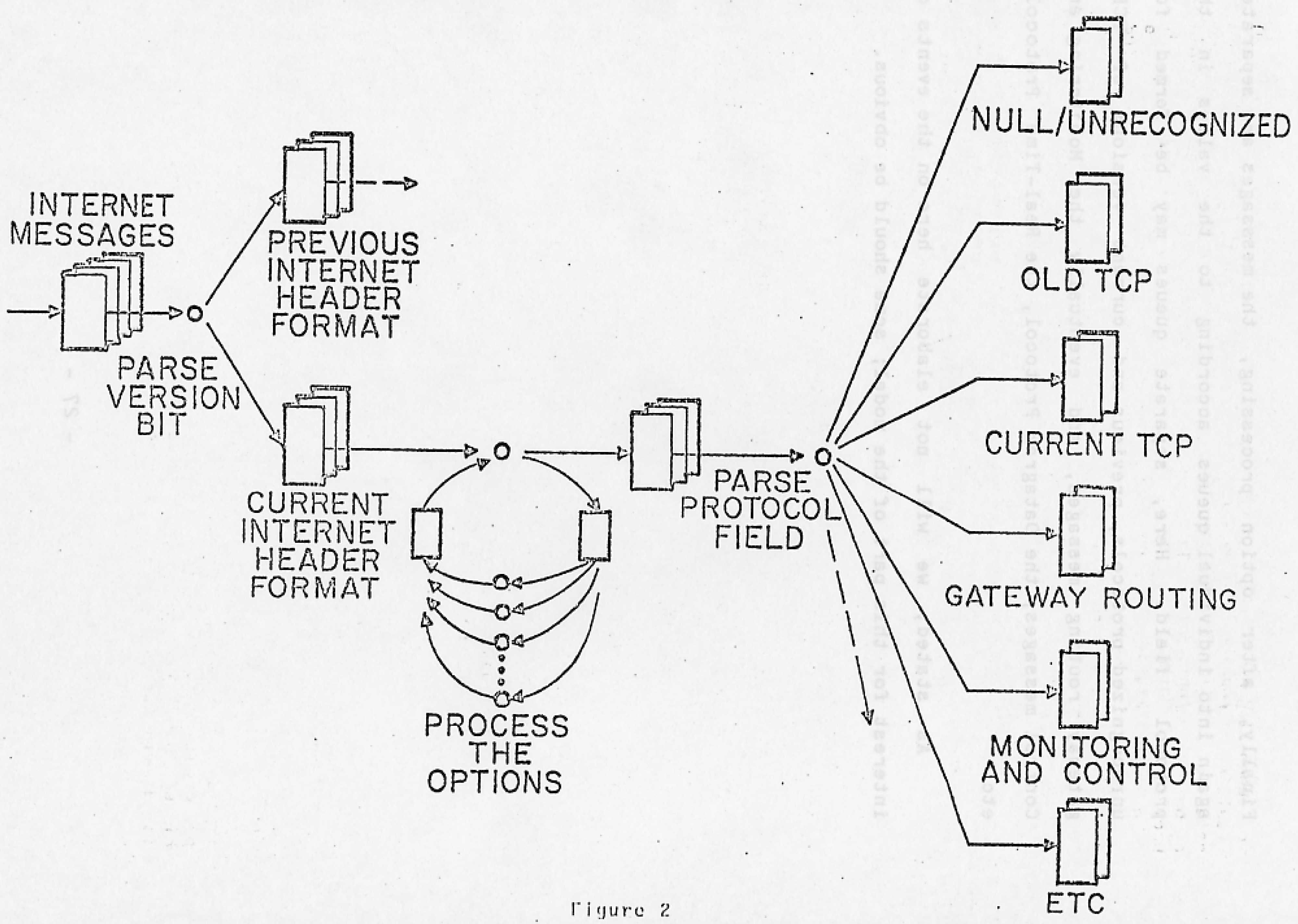

Figure 2

Finally, after option processing, the messsages are separated again into individual queues according to the values in the protocol field. Here, separate queues may be formed for unrecognized protocols, previous and current versions of TCP, gateway routing messages, and eventually the Monitoring and Control messages, the Datagram Protocol, the Real-Time Protocol, etc.

As stated, we will not elaborate here on the events of interest for this part of the model; some should be obvious.

5.5 Unresolved Issues

In this section we want to address a number of issues which are not yet resolved in the modelling of the gateway component. Their resolution will probably depend on prolonged discussions in certain cases, on snap decisions in others; possibly in some cases a satisfactory resolution will not be possible, and whatever alternative solutions are proposed will all have to be included in a generalized modelling to make sure the modelling is comprehensive enough.

At any rate, the topics which need further investigation are presented below.

5.5.1 Are the Event Types Correct

First there needs to be some general agreement that the event types (information types) we have identified are sufficient for modelling purposes. It is probably the case that they are correct enough for a beginning, and that no particularly disruptive perturbation of the model would be caused if another event type had to someday be accommodated or an existing event type had to be deleted. However, the omission of an XNET information type (not to be confused with the distinct "debugging" type) may have to be remedied before too long.

5.5.2 What Information Should be Communicated to the GMCC

Here there is a lot of room for varying opinion. The next cut at the model will try to identify as many potential events of interest as possible. Obviously, not all these events will be of interest to all implementers; that's why the Monitoring and Control mechanisms must be sure to allow for only partial participation on the part of any particular implementation. However, it may also be the case that we omit some event that is of particular interest to some gateway implementer, or the information which we choose to record about the event is not quite what is needed for some implementer's needs. Our collection mechanism must be flexible enough to accommodate extensions at any time.

Here the real issue may be how to control, administratively, the selection of events of interest so that all parties are satisfied with the set selected.

5.5.3 How Should Information be Communicated to the GMCC

We are of the opinion that in most cases, the basic gateway-to-GMCC communication facility should be datagram oriented on the basis that (1) connection overhead may be too expensive for certain gateway implementations, that (2) no flow control is probably needed since the gateways will not be generating data too frequently and since GMCCs will generally have substantially greater storage and processing capabilities than will the gateways, and that (3) a practical reporting scheme can probably be developed in which lost messages will not necessarily represent lost information, merely delayed delivery of information, since the contents of lost messages can be inferred from later messages, (this is the case for cumulative statistics for example); on the other hand, the datagrams should carry sequencing information, and will of course employ standard Internet Headers.

Datagram service will be satisfactory in most cases, we hope. In certain instances, however, reliable transmissions may be extremely important; for these instances, some additional capability may have to be added. As yet, we have no real feel for the capabilities required; thus this is still an open issue. Also at issue is the decision as to whether internet messages directed at the various gateway processes should all carry a single protocol designator, or whether each different message type should command a distinct designator. It is not yet clear whether minimal gateway implementations would find it easier to process messages formatted in one way vs. the other, or whether it is too wasteful of the Internet Header's protocol field, or whether it is easier one way or the other to subsequently add or delete new message formats.

Beyond these basic issues, there is also the issue of message formats and message content. Two alternatives present themselves as regards event reporting. We assume that event counters can be maintained in 16-bit words, say. We can insist that messages contain a fixed number of counters in a fixed order, and thus eliminate the need to transmit descriptive information with each reporting message. Or, we can allow for every counter to be accompanied by another word which names the counter. Tradeoffs between the two strategies are not yet properly understood.

5.5.4 Addressing Processes from Outside the Gateway

Each of the gateway processes responsible for some activity of Monitoring and Control has a unique identity, or name, within the Catenet. But because the gateway is attached to multiple nets, it is possible for each process to have multiple distinct addresses. We can assume that one reasonable modelling for the net-dependent input handler requires the input handler to recognize at the net interface all the messages addressed to processes which share its own net (and host) address. This is the case for example in a general purpose implementation of the gateway, since the general purpose input handler doesn't normally receive messages for processes that don't share its own net address. It probably should not be the responsibility of a given net-dependent input handler to be aware that it is playing a role in a gateway implementation, and thus to be cognizant of the alternative internet addresses of the gateway processes it thinks of as its own; i.e. the net-dependent code should not have to know what other nets the gateway is connected to. Therefore, if a message arrives at one net interface that specifies a resource (process) whose net address is different from that of the input handler, then the message should simply be handed off to a special process which can effect the proper disposition for the message without further involvement of the input handler.

Figure 3 depicts this arrangement.

Here, each network has its own interface to a common copy of the gateway process, so that it can communicate with it directly whenever a message arrives which addresses the process via the appropriate net. However, when a message is received for a destination not known to the input handler, the message is simply handed to the special process, which in this figure is referred to as the "Postman". Note that the Postman can effect delivery to the processes via its own interface, so that successful delivery does not depend on the route taken by the message. (Note that the model might want to specify that the Postman be able to add messages to the input queues of the net-dependent input handlers as a means for effecting delivery in a uniform way.) The Postman here also has the responsibility for performing the tasks associated with the routing of messages to distant locations. That is, messages input at the gateway which are only passing through should be routed by the general gateway routing algorithms; these can be implemented by the Postman, or by some other process to which the Postman interfaces.

There is, of course, another way to model the net-dependent/net-independent boundary. Figure 4 shows this other way.

MODEL 1

EACH NET RECOGNIZES ONE ADDRESS
FOR EACH PROCESS

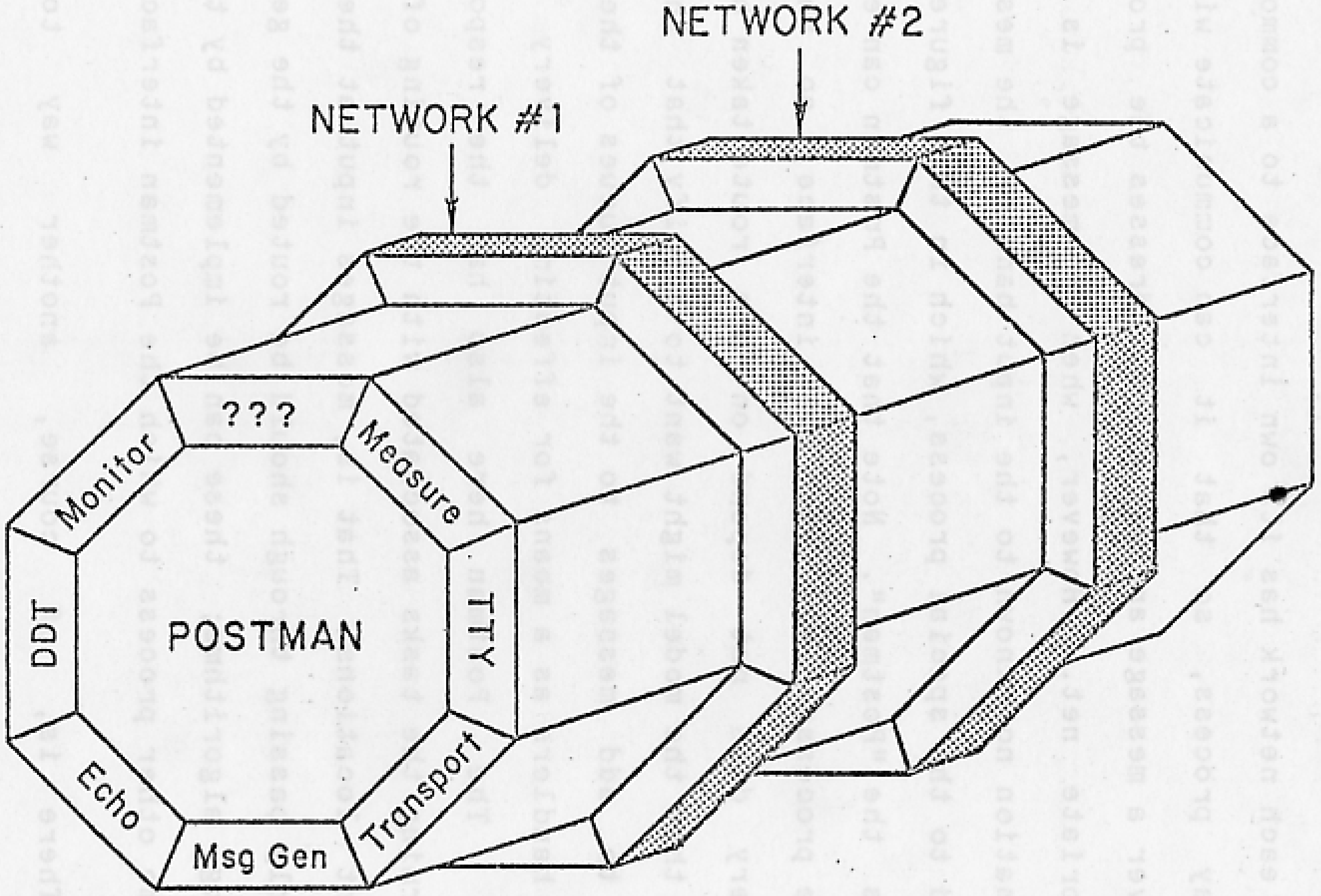

Figure 3

MODEL 2
EACH NET PASSES ALL MESSAGES TO THE POSTMAN

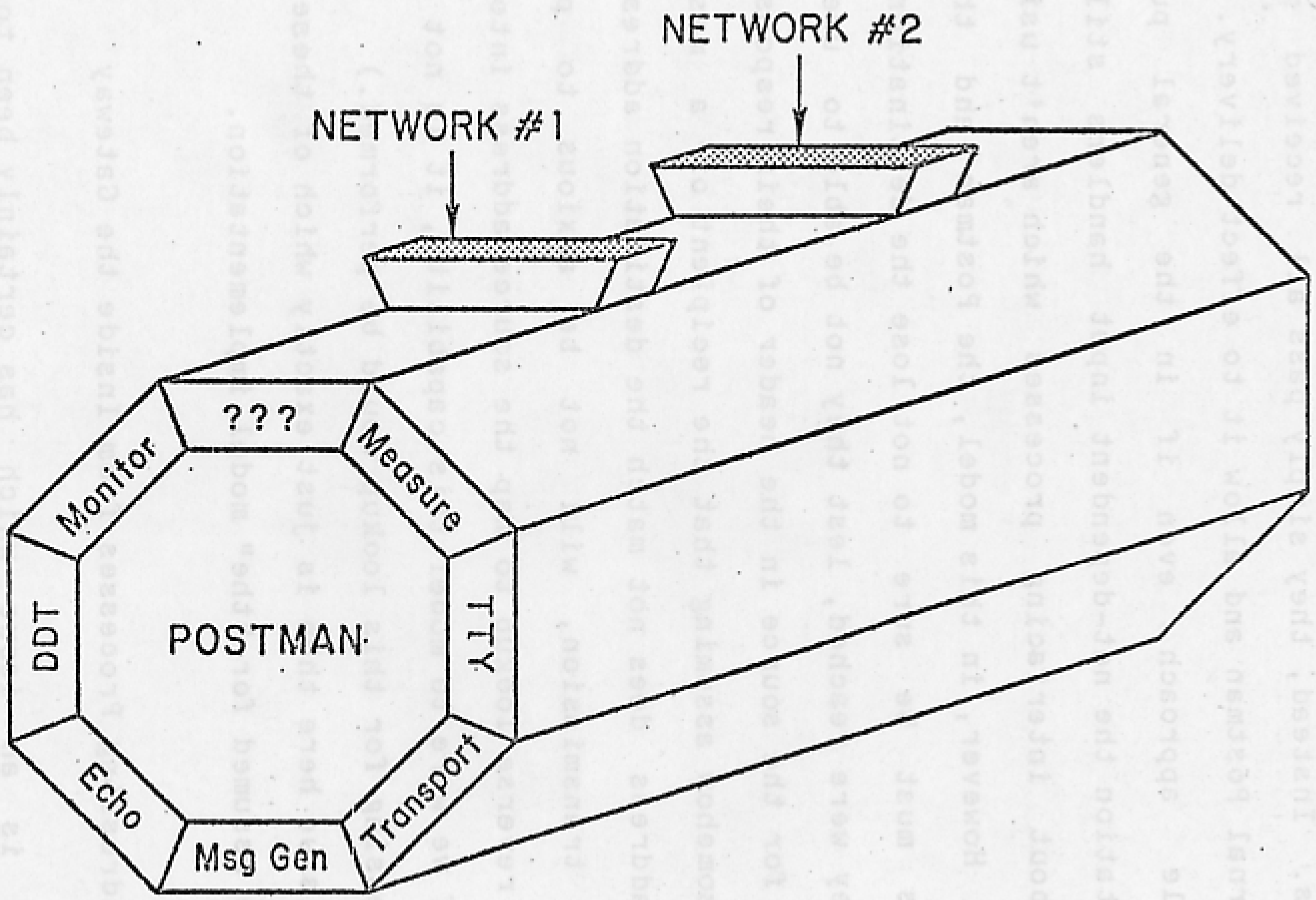

Figure 4

Basically the difference here is that the net-dependent input handlers no longer have their own interfaces to the gateway processes. Instead, they simply pass all received messages to the internal Postman and allow it to effect delivery. This is an acceptable approach even if in the general purpose host implementation the net-dependent input handlers still have to worry about interfacing processes which aren't using Internet Headers. However, in this model, the Postman and the affected processes must be sure to not lose the destination address by which they were reached, lest they not be able to use the same address for the source in the header of their response messages. (We are somehow assuming that the recipient of a message whose source address does not match the destination address which was used for transmission, will not be anxious to perform the required reverse lookup to map the source address into a resource name. If we were to model this capability, it is not clear where the processing for this lookup would be performed.)

At issue here then is just exactly which of these two models should be assumed for "the" model implementation.

5.5.5 Addressing Processes from Inside the Gateway

Here is an issue which has certainly been touched on in other internet meetings; it is basically a discussion of the need for high level protocols to carry their own addressing information.

Gateway processes will have occassion to communicate with other processes in support of gateway routing or gateway Monitoring and Control. Traffic between two gateway processes may be intrahost, intranet, or internet, depending on the relative locations of the source and destination processes. At issue is whether in all cases a single data transport protocol should be used to effect message delivery. We have already "concluded" in the discussion of event reporting that gateway Monitoring and Control messages should employ Internet Headers in all cases. And it would certainly seem on the surface that this scheme is ideal. However, in certain cases this may not be true.

We are struck by an inconsistency which arises when we try to attain symmetry in modelling the gateway's internal organization. At one point in the model, we have a process whose job it is to route messages through a given local net. Whenever it is handed an internet message, it analyzes the header of the message in order to determine the desired destination, and hence what local address to specify in the net-dependent data transport protocol.

The inconsistency is found because we don't have a corresponding process whose job it is to analyze higher level protocol headers in order to route messages through the internet (using the internet data transport protocol). This means either that each of our Monitoring and Control processes must make up an Internet Header itself, or that, if some common process is to do it, the common process must be handed the addressing information by some "out-of-band" path (such as a shared memory control block). This may not be easy to provide for a distributed gateway implementation, say.

The real issue here, of course, is inspired by the problems we see for, say, TCP users, if the Internet and TCP Headers recently proposed are adopted for use in the Catenet. The fact that higher level protocols are being designed which don't carry their own addressing information means that these protocols will be practically unusable in any instance where the data transport protocol used to carry the messages is different from the data transport protocol embodied in the Internet Header. TCP, for example, would probably not be usable without Internet Headers in the ARPANET, since port addressing would be impossible.

Although it is probably the case that we will not opt to include addressing information in the messages which adhere to our higher level Monitoring and Control protocols, and will thus in fact choose to use the Internet Header to provide addressing, we nevertheless wonder if it is wise to arbitrarily restrict these protocols to use with only a single data transport protocol.